New York University Tandon School of Engineering
Computer Science and Engineering

CS-GY 6923: Midterm Exam Practice.

Directions

- This is a practice exam. It is not graded. It is intended to help you prepare for the midterm exam. The questions are representative of the types of questions that *might* be on the exam, but the actual exam may differ in format and content.

1. Always, Sometimes, Never. (12pts – 3pts each)

Indicate whether each of the following statements is ALWAYS true, SOMETIMES true, or NEVER true. **No justification is necessary to receive full credit for a correct answer**. To earn partial credit if you are wrong, you may provide a short justification or example to explain your choice.

(a) The empirical risk of a model is lower than the population risk.

ALWAYS **SOMETIMES** NEVER

Empirical risk is a random quantity which depends on the data. Even when it's equal to the population risk in expectation, it could be larger or smaller depending on chance.

(b) When minimizing a loss function $L(\beta)$ using gradient descent, decreasing the learning rate η slows down how quickly we converge to a local minimum.

ALWAYS SOMETIMES NEVER If the learning rate is too small, it will take a long time to reach the local minimum. If the learning rate is too big, the loss may bounce around and not reach the local minimum.

(c) For two random events X and Y, $\Pr(X \mid Y) \cdot \Pr(Y) = \Pr(Y \mid X) \cdot \Pr(X)$.

ALWAYS SOMETIMES NEVER

This is the definition of conditional probability.

(d) Consider a multiple linear regression problem where each data example has the form $(\vec{x}, y) = ([x_1, x_2], y)$. Transform the predictor variables by adding quadratic terms, so each new data example has the form $(\vec{x}_{trans}, y) = ([x_1, x_2, x_1^2, x_2^2, x_1 x_2], y)$. Let L^* be the minimum training loss for the original problem and let L^*_{trans} be the minimum training loss for the transformed problem. Is $L^*_{trans} \leq L^*$?

ALWAYS SOMETIMES NEVER

For any $\vec{\beta} = [\beta_1, \beta_2]$, $L_{trans}([\beta_1, \beta_2, 0, 0, 0]) = L([\beta_1, \beta_2])$. So it is always possible to find parameters that ensure $L_{trans} \leq L$. It must therefore be that $\min L_{trans} \leq \min L$.

2. Model Diagnosis Short Answer (8pts)

You are trying to solve a prediction problem using a multiple linear regression model with ℓ_2 loss. You first split the data set into a train set (80%) and a test set (20%). You then train the model on the train set to obtain a parameter vector $\vec{\beta}$. Using $\vec{\beta}$, you evaluate the average squared loss of the regression model on the train and test set, separately.

For each of the following scenarios, circle all answers that apply. **No justification is necessary to receive full credit for a correct answer**. To earn partial credit if you are wrong, you may provide a short justification.

(a) (4pts) The average squared loss on the train set is 1.5 and the average squared loss on the test set is 12.6. **Which of the following techniques is likely to improve your average test loss?**

REGULARIZATION **FEATURE SELECTION** FEATURE TRANSFORM DATA SCALING

We appear to be overfitting since our train loss is much less than our test loss. Regularization or feature selection would be the most appropriate cures. Feature transformation typically leads to a richer model, so only makes overfitting worse. And data scaling doesn't do anything for linear regression!

(b) (4pts) The average squared loss on the train set is 10.2 and the average squared loss on the test set is 9.9. **Which of the following techniques is likely to improve your average test loss?**

REGULARIZATION FEATURE SELECTION **FEATURE TRANSFORM** DATA SCALING

We appear to be performing poorly on both train and test loss, so we might need a richer model. The best cure would therefore be some sort of feature transformation. Of course, it's not guaranteed to work, but would be worth a try.

3. Convexity of Regularized Regression (6pts)

In the lecture notes, we proved that the least squares regression loss $L(\beta) = \|\mathbf{X}\beta - \mathbf{y}\|_2^2$ is convex. It is also possible to show that the ℓ_2 and ℓ_1 penalty functions, $g(\boldsymbol{\beta}) = c\|\boldsymbol{\beta}\|_2^2$ and $g(\boldsymbol{\beta}) = c\|\boldsymbol{\beta}\|_1$, are convex. In this problem you will prove a statement which implies that, when these penalty functions are added to the standard regression loss, the resulting regularized loss is also convex.

To show that the function $h\boldsymbol{\beta} = f\boldsymbol{\beta} + g\boldsymbol{\beta}$ is convex, given that f and g are convex, we need to show that for any two points $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \mathbb{R}^d$ and $\lambda \in [0, 1]$ the following inequality holds:

$$h(\lambda\boldsymbol{\beta}_1 + (1-\lambda)\boldsymbol{\beta}_2) \leq \lambda h(\boldsymbol{\beta}_1) + (1-\lambda)h(\boldsymbol{\beta}_2) \tag{1}$$

Given that f and g are convex, by definition we have that for any $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \mathbb{R}^d$ and $\lambda \in [0, 1]$ the convexity inequalities hold:

$$f(\lambda\boldsymbol{\beta}_1 + (1-\lambda)\boldsymbol{\beta}_2) \leq \lambda f(\boldsymbol{\beta}_1) + (1-\lambda)f(\boldsymbol{\beta}_2)$$
$$g(\lambda\boldsymbol{\beta}_1 + (1-\lambda)\boldsymbol{\beta}_2) \leq \lambda g(\boldsymbol{\beta}_1) + (1-\lambda)g(\boldsymbol{\beta}_2)$$

Adding these two inequalities together, we get:

$$f(\lambda\boldsymbol{\beta}_1 + (1-\lambda)\boldsymbol{\beta}_2) + g(\lambda\boldsymbol{\beta}_1 + (1-\lambda)\boldsymbol{\beta}_2) \leq \lambda f(\boldsymbol{\beta}_1) + (1-\lambda)f(\boldsymbol{\beta}_2) + \lambda g(\boldsymbol{\beta}_1) + (1-\lambda)g(\boldsymbol{\beta}_2)$$

This simplifies to:

$$h(\lambda\boldsymbol{\beta}_1 + (1-\lambda)\boldsymbol{\beta}_2) \leq \lambda h(\boldsymbol{\beta}_1) + (1-\lambda)h(\boldsymbol{\beta}_2) \tag{2}$$

Since equation (2) is the same as (1), we have shown that the regularized loss is convex.

4. Bayesian Crab Classification (10pts)

A biologist is collecting specimens from two species of crabs, species S_0 and S_1. These species live in the same habitat and look similar to the human eye. To accelerate crab sorting by species, the biologist wants to develop a simple classification rule based on body measurements. She observes that the ratio of *forehead*

breadth to overall *body length* differs between crabs in species S_0 and S_1. The biologist proposes to measure this ratio (denoted by R) and use it as a single predictor variable for classification.

The biologist assumes that the crab data comes from a "mixture of Gaussians" probabilistic model. In particular, she assumes that for each species, R follows a normal (Gaussian) probability distribution, with different parameters for each species. The biologist makes the following concrete observations:

- 35% of all crabs collected belong to S_0 and the remaining 65% belong to S_1.
- For crabs in S_0, the average value of R is .5. For crabs in S_1, the average value of R is .4.
- For both species, the standard deviation of R is .1.

(a) (6pts) Suppose we collect a new crab with forehead breadth to body length ratio R_{new}. The biologist would like to assign this crab to S_0 or S_1 using a maximum a posterior (MAP) classification rule. Denote this rule by $f : \mathbb{R} \to \{S_0, S_1\}$. The rule takes as input the ratio R_{new} and outputs S_0 or S_1.

Write down all mathematical expressions that would need to be evaluated to compute f for a given input R_{new}. Your expressions do not need to be simplified, but they should not involve unknown variables besides R_{new}. **Hint:** Use Bayes rule.

To implement a MAP estimator we need to compute:

$$\Pr(S_0 \mid R_{new}) \qquad \text{and} \qquad \Pr(S_1 \mid R_{new})$$

Using Bayes rule we have:

$$\Pr(S_0 \mid R_{new}) = \frac{\Pr(R_{new} \mid S_0)\Pr(S_0)}{\Pr(R_{new})} \qquad \text{and} \qquad \Pr(S_1 \mid R_{new}) = \frac{\Pr(R_{new} \mid S_1)\Pr(S_1)}{\Pr(R_{new})}$$

Note that, we have:

$$\Pr(S_0) = .35 \qquad \text{and} \qquad \Pr(S_1) = .65.$$

And using the equation for the Gaussian distribution, we also have:

$$\Pr(R_{new} \mid S_0) = \frac{1}{\sqrt{2\pi}\cdot .1} e^{-\frac{(R_{new}-.5)^2}{2\cdot .1^2}} \qquad \text{and} \qquad \Pr(R_{new} \mid S_1) = \frac{1}{\sqrt{2\pi}\cdot .1} e^{-\frac{(R_{new}-.4)^2}{2\cdot .1^2}}.$$

(b) (4pts) Show that, for this problem, the classification rule f has the following form:

$$f(R_{new}) = \begin{cases} S_0 \text{ if } R_{new} \geq \lambda \\ S_1 \text{ if } R_{new} < \lambda, \end{cases}$$

for some fixed threshold parameter λ (you do not need to explicitly compute λ).

Note: A less formal argument than what I give below would suffice.

The MAP classification rule f is as follows:

$$f(R_{new}) = \begin{cases} S_0 \text{ if } \Pr(S_0 \mid R_{new}) \geq \Pr(S_1 \mid R_{new}), \\ S_1 \text{ if } \Pr(S_0 \mid R_{new}) < \Pr(S_1 \mid R_{new}). \end{cases}$$

Substituting in our equations from part (a), we see that we will classify a crab with ratio R_{new} into class S_0 under this rule if:

$$f(R_{new}) = S_0 \text{ if } \qquad \frac{.35\frac{1}{\sqrt{2\pi}\cdot .1} e^{-\frac{(R_{new}-.5)^2}{.02}}}{\Pr(R_{new})} \geq \frac{.65\frac{1}{\sqrt{2\pi}\cdot .1} e^{-\frac{(R_{new}-.4)^2}{.02}}}{\Pr(R_{new})}$$

which is equivalent to checking if

$$f(R_{new}) = S_0 \text{ if } \qquad .35e^{-\frac{(R_{new}-.5)^2}{.02}} \geq .65e^{-\frac{(R_{new}-.4)^2}{.02}}.$$

Taking logs and rearranging, this in turn is equivalent to checking:

$$f(R_{new}) = S_0 \text{ if } \qquad (R_{new}-.4)^2 - (R_{new}-.5)^2 \geq .02\log(.65) - .02\log(.35)$$
$$f(R_{new}) = S_0 \text{ if } \qquad .2 \cdot R_{new} \geq .02\log(.65) - .02\log(.35) - .16 + .25$$

So clearly we classify in S_0 if $R_{new} \geq \lambda$ for some λ.

5. Loss Minimization. (10pts)

For data with one predictor and one target: $(x_1, y_1), \ldots, (x_n, y_n)$, consider a linear regression model:

$$f_{\beta_0,\beta_1}(x) = \beta_0 + \beta_1 x$$

with *exponential loss*:

$$L(\beta_0, \beta_1) = \sum_{i=1}^{n} e^{(y_i - f_{\beta_0,\beta_1}(x_i))^2}$$

(a) (5pts) Write down an expression for the gradient of the loss L.

We can compute both partial derivatives to get the gradient. For both we need to use chain rule:

$$\frac{\partial L}{\partial \beta_0} = \sum_{i=1}^{n} -2 \cdot (y_i - \beta_0 - \beta_1 x_i) \cdot e^{(y_i - \beta_0 - \beta_1 x_i)^2}$$

$$\frac{\partial L}{\partial \beta_1} = \sum_{i=1}^{n} -2x_i \cdot (y_i - \beta_0 - \beta_1 x_i) \cdot e^{(y_i - \beta_0 - \beta_1 x_i)^2}$$

And then we have $\nabla L(\beta_0, \beta_1) = \begin{bmatrix} \partial L/\partial \beta_0 \\ \partial L/\partial \beta_1 \end{bmatrix}$.

If you wanted to do everything in matrix form, let $\mathbf{X}$ be a data matrix with first column containing $x_1, \ldots, x_n$ and second column containing all 1s. Then let $\vec{w} = (\vec{y} - \mathbf{X}\vec{\beta}) \cdot exp(\vec{y} - \mathbf{X}\vec{\beta})$ where all operations are applied entrywise. Then the gradient is $\mathbf{X}^T\vec{w}$.

(b) (2pts) Name two algorithms/methods which could be used to minimize L.

- Brute force search.
- Gradient descent.

(c) (3pts) In general, is this exponential loss more or less robust to outliers when compared to ℓ_2 loss? How about when compared to ℓ_∞ loss?

If the linear model under exponential loss is effected *more* by an outlier, we say the loss is *less* robust.

It is therefore less robust than ℓ_2 loss because the loss function punishes outliers significantly more: you pay a cost of $e^{\text{squared error}}$ instead of just the square error. It will thus try to more closely fit outliers. On the other hand, the exponential loss is more robust than ℓ_∞ loss, which punishes outliers pretty much as much as possible: we only care about the loss of the worst data example.

Table 1: Bayes Counts

w	P(w | C_1)	P(w | C_2)
machine	3/11	1/5
learning	3/11	3/10
good	4/11	1/5
evil	1/11	3/10

6. Naive Bayes. (10pts)

Consider the following spam detection dataset.

1. (doc1) machine learning good (C1)
2. (doc2) machine good (C1)
3. (doc3) learning good (C1)
4. (doc4) evil learning good (C2)
5. (doc5) machine learning evil (C2)

The dataset consists of 5 documents, each of which is labeled as either "C1" or "C2". The class of each document is indicated in parentheses. Use the Naive Bayes classifier to classify the following document as either "C1" or "C2". New document: **"good evil machine"**. Use add1 smoothing. Show your work.

SOLUTION: The Table with the counts is in 1.
Priors probabilities: $\Pr(C_1) = \frac{3}{5}$ and $\Pr(C_2) = \frac{2}{5}$.
Solution:

$$\Pr(C_1|good, evil, machine) = \Pr(good, evil, machine|C_1) \cdot \Pr(C_1) = \frac{1}{11} \cdot \frac{1}{11} \cdot \frac{3}{11} \cdot \frac{3}{5}$$
$$= \frac{36}{6655} = 0.0054$$

$$\Pr(C_2|good, evil, machine) = \Pr(good, evil, machine|C_2) \cdot \Pr(C_2) = \frac{1}{5} \cdot \frac{3}{10} \cdot \frac{1}{5} \cdot \frac{2}{5}$$
$$= \frac{6}{1250} = 0.0048$$

Since C1 > C2 the document is classified as C1.

7. Decision Tree Learning. (10pts)

Given the following table of data,

(a) (6pts) Construct a decision tree that classifies the data. Use Information Gain as the splitting criterion and show your work.

(b) (4pts) What is your prediction for D15?

Let H(D) be the entropy of the dataset D. Since we have two classes, and 14 training samples, 9 of them are yes and 5 of them are no. Then we have:

$$H(D) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

Day	Outlook	Humidity	Wind	Play
D1	sunny	high	weak	no
D2	sunny	high	strong	no
D3	overcast	high	weak	yes
D4	rain	high	weak	yes
D5	rain	normal	weak	yes
D6	rain	normal	strong	no
D7	overcast	normal	strong	yes
D8	sunny	high	weak	no
D9	sunny	normal	weak	yes
D10	rain	normal	weak	yes
D11	sunny	normal	strong	yes
D12	overcast	high	strong	yes
D13	overcast	normal	weak	yes
D14	rain	high	strong	No
D15	rain	high	weak	???

Next we need to compute the information gain for each attribute A, where A is either Outlook, Humidity, or Wind.

$$\begin{aligned} IG(D, \text{Outlook}) &= H(D) - \frac{5}{14}H(D_1) - \frac{4}{14}H(D_2) - \frac{5}{14}H(D_3) \\ &= 0.94 - \frac{5}{14} \cdot 0.971 - \frac{4}{14} \cdot 0.811 - \frac{5}{14} \cdot 0.971 \\ &= 0.246 \end{aligned}$$

$$\begin{aligned} IG(D, \text{Humidity}) &= H(D) - \frac{7}{14}H(D_4) - \frac{7}{14}H(D_5) \\ &= 0.94 - \frac{7}{14} \cdot 0.985 - \frac{7}{14} \cdot 0.985 \\ &= 0.151 \end{aligned}$$

$$\begin{aligned} IG(D, \text{Wind}) &= H(D) - \frac{8}{14}H(D_6) - \frac{6}{14}H(D_7) \\ &= 0.94 - \frac{8}{14} \cdot 0.811 - \frac{6}{14} \cdot 0.918 \\ &= 0.048 \end{aligned}$$

As we can see outlook is the winner, so we split on outlook. Looking at the outlook column, we see that there are 5 sunny days, 4 overcast days, and 5 rainy days. We can then split the dataset into three subsets, D1, D2, and D3, where D1 is the sunny days, D2 is the overcast days, and D3 is the rainy days. Looking at the entropy of each subset, we see that we have:

$$\begin{aligned} H(D_1) &= -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971 \\ H(D_2) &= -\frac{4}{4}\log_2\frac{4}{4} = 0 \\ H(D_3) &= -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971 \end{aligned}$$

Pictorially, we can see the tree structure for outlook in 1. The tree for the other two attributes are similar, and the full tree is shown in 2.

8. Reporting. (10pts)

Consider the following table:

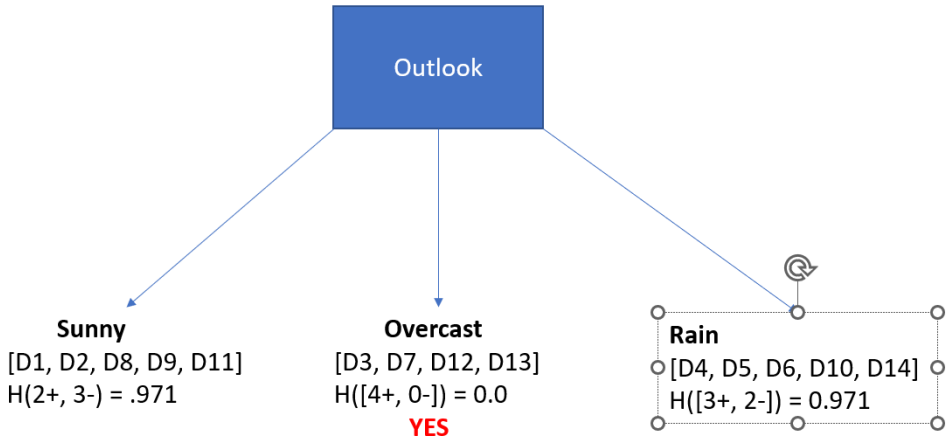

Figure 1: Outlook subtree

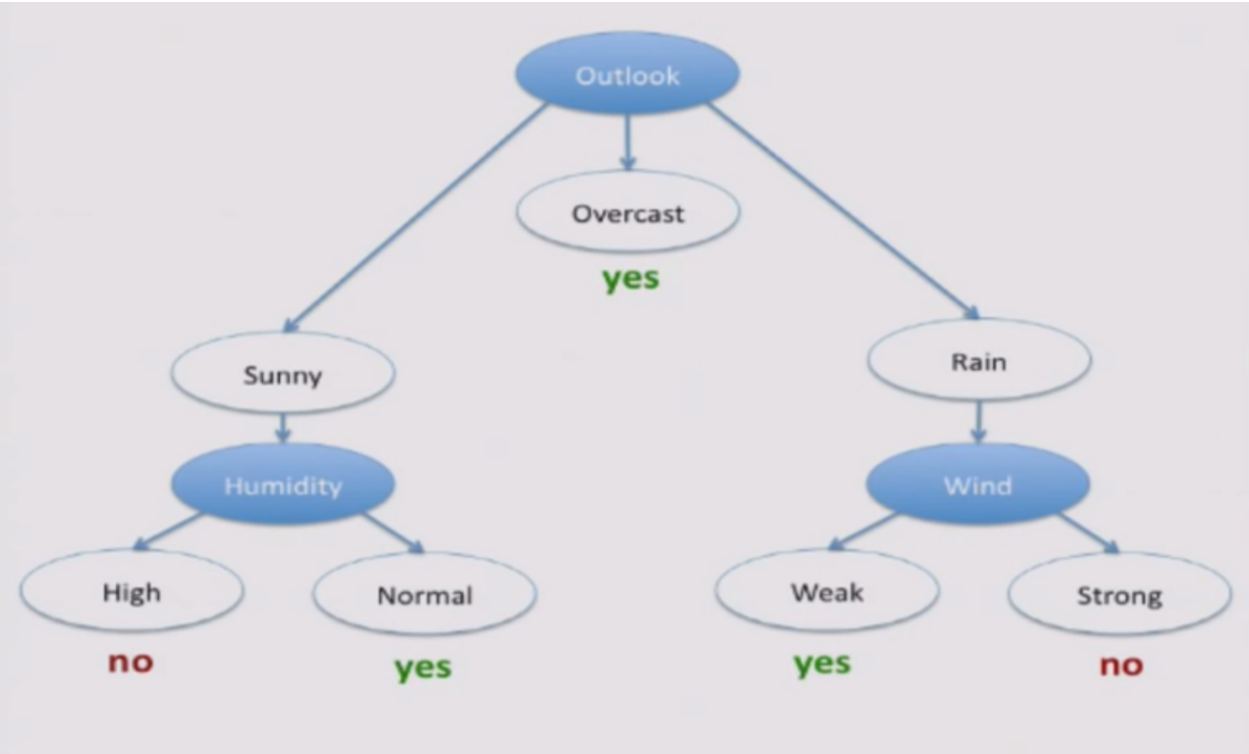

Figure 2: Full tree

Individual Number	1	2	3	4	5	6	7	8	9	10	11	12
Actual Classification	1	1	1	1	1	1	1	1	0	0	0	0
Predicted Classification	0	0	1	1	1	1	1	1	1	0	0	0

(a) (2pts) What is the accuracy of the classifier?

(b) (2pts) What is the precision of the classifier?

(c) (2pts) What is the recall of the classifier?

(d) (2pts) What is the F1 score of the classifier?

(e) (2pts) What is the confusion matrix of the classifier?

SOLUTION: We can perform the comparison between actual and predicted classifications and add this information to the table, making correct results appear in green so they are more easily identifiable. We can see the resuls in Table 2.

Table 2: Confusion Matrix Results

Individual Number	1	2	3	4	5	6	7	8	9	10	11	12
Actual Classification	1	1	1	1	1	1	1	1	0	0	0	0
Predicted Classification	0	0	1	1	1	1	1	1	1	0	0	0
Result	FN	FN	TP	TP	TP	TP	TP	TP	FP	TN	TN	TN

(a) (2pts) What is the accuracy of the classifier? The accuracy is the number of correct results divided by the total number of results. In our case, we have 9 correct results and 3 incorrect results, so the accuracy is $9/12 = 0.75$.

(b) (2pts) What is the precision of the classifier? The precision is the number of true positives divided by the number of true positives plus the number of false positives. In our case, we have 6 true positives and 1 false positives, so the precision is $6/7 = 0.857$.

(c) (2pts) What is the recall of the classifier? The recall is the number of true positives divided by the number of false negatives. In our case, we have 6 true positives and 2 false negatives, so the recall is $6/8 = 0.75$.

(d) (2pts) What is the F1 score of the classifier? The F1 score is the harmonic mean of the precision and recall. In our case, the F1 score is $2 \cdot \frac{0.857 \cdot 0.75}{0.857+0.75} = 0.8$.

(e) (2pts) What is the confusion matrix of the classifier? The confusion matrix is a table that shows the number of true positives, false positives, true negatives, and false negatives. In our case, we have 6 true positives, 1 false positive, 2 false negatives, and 3 true negatives. The confusion matrix is shown in Table 3.

Table 3: Confusion Matrix

	Predicted Positive	**Predicted Negative**	**Total**
Actual Positive	6	2	8
Actual Negative	1	3	4
Total	7	5	12

9. Practicum. (10pts)

Write a python function: `K_Fold_Cross_Validation(X, y, K, model)` that takes as input a dataset X and labels y, a number of folds K, and a model *model*. The function should return the average accuracy of the model on the dataset.

You may use the following functions:

(a) `model.fit(X, y)`. Fits the training data with model order p: $\hat{B} = fit(X, y, p)$.

(b) `model.predict(X)`. Predicts the values on the test data.

(c) `model.score(X, y)`. Returns the accuracy of the model on the test data.

SOLUTION:
One potential solution is shown below.

```
def K_Fold_Cross_Validation(X, y, K):

        n = X.shape[0]  # Get number of samples
        indices = np.arange(n)  # Get indices of samples
        np.random.shuffle(indices)  # Shuffle indices
        indices = np.array_split(indices, K)  # Split indices into K groups
        accuracies = []  # Initialize accuracies list
        for i in range(K):  # Loop through each K fold
                # Get test indices
                test_indices = indices[i]
                # Get training indices
                train_indices = np.concatenate(indices[:i] + indices[i+1:])
                # Get training data
                X_train = X[train_indices]
                y_train = y[train_indices]
                # Get test data
                X_test = X[test_indices]
                y_test = y[test_indices]
                # Fit model
                model.fit(X_train, y_train)
                # Get accuracy
                accuracy = model.score(X_test, y_test)
                # Add accuracy to list
                accuracies.append(accuracy)

        # Return average accuracy
        return np.mean(accuracies)
```